VSP International
Science Publishers
P.O. Box 346, 3700 AH Zeist
The Netherlands

Lecture Series on Computer and Computational Sciences
Volume , 2006, pp. -

Web Data Extraction using Semantic Generators

David Camacho and Maria D. R-Moreno[1]

Computer Science Department. Universidad Autónoma de Madrid.
Madrid, Spain. E-mail:david.camacho@uam.es

Departamento de Automática. Universidad de Alcalá. Alcalá de Henares (Madrid), Spain.
E-mail: mdolores@aut.uah.es

Abstract: The faster growing in both, contents and formats, of the World Wide Web make really difficult to use the available information stored in millions of servers. Information Extraction provide a set of techniques to help in the process of identify and retrieve this information. In this paper, we propose an approach to extract information from HTML pages and to add semantic (in form of XML tags) to the data. This is achieved by helping a user by means of an assistant tool called WebMantic. The interaction with the user is used to acquire the semantic XML tag and to build several rules (called Semantic Generators) that will be used later in the extraction process.

Keywords: Information Extraction, Wrappers generation, Web data processing, XML.

1 Introduction

Currently the Web provides a huge amount of unstructured and non-semantic information available for both, users, and automatic crawler programs. Although the Web is evolving to build sites with structured and semantic information, these new kind of Web sites are meant to be deployed in business-to-business (B2B) scenarios. Therefore, most of users (and Web applications) will continue to access data in HTML format [1]. To build systems able to access and extract the information stored in the HTML sources, wrappers are commonly used. Wrappers are specialized programs that automatically extract data from documents and convert the information stored into a structured format. Three main functions need to be implemented in a Wrapper: First, they must be able to download HTML pages from a website. Second, they must search for, recognize, and extract the specified data. Third, they have to save this data in a suitably structured format to enable further manipulation. XML is very interesting to structure information, as there are many tools that can use it (like XPath). Several research fields like: Information Gathering, Information Extraction, or Web Mining, have been involved in the process of extracting and managing the information stored in the the Web. [3, 6]

This paper presents both, a general approach based on rules, that can be used to automatically generate wrappers, and an assistant generator wrapper (called WebMantic) that builds the wrapper. Our approach allows to create wrappers that obtain XML documents from HTML pages. We have defined a flexible filtering and preprocessing technique that allows to translate some pieces of information, that contain the desired information, into a more understandable and semantic representation. Only the required portions of the page will be translated. Non-selected parts of

[1]Corresponding author.

the HTML document will be ignored. Finally, we will work under a reasonable limitation: only structured information, like data stored in lists or tables, will be considered.

Our technique has some similarities and differences with other related wrapper/extraction techniques. The interaction with the user is necessary to identify the information to be extracted, like in other approaches (i.e. Lixto [2]). However, the user in our approach is used for both, to select the information to be retrieved, and to provide the semantics of this information. The HTML preprocessing task is sequentially performed and the user decides what is the information that will be transformed, and define the XML tags (i.e. the semantics) that will be finally used to transform the document. Our technique is agent-based oriented, therefore these wrappers must to be integrated into an information agent to be executed. Other approaches, such as or Wien [5], use Machine Learning algorithms to learn automatically the wrappers, the representation used by the inductive algorithms usually is too restrictive to wrap complex semistructured pages with variant structuring patterns.Other systems such us proposed by Hsu et al. [4] relies on *heuristics* to guide the implementation of the wrappers,these approaches only cover a small proportion of the Web pages, and it is difficult to extend their heuristics to other kind of Web pages. Currently it is possible to find more than thirty tools to aid in the wrapper generation process[2].

This paper is structured as follows. In Section 2, the rules that are used to translate HTML documents into XML documents (named Semantic Generator), are described. In Section 3 our approach to the problem of wrapper generation, called WebMantic, is briefly shown. Finally, Section 4 summarizes the conclusions of the paper.

2 The Semantic Generators

We define a Semantic Generator, or S_g, as a set of rules ($HTML_2XML$), that can be used to translate HTML documents into XML documents. A Semantic Generator (S_g), is built by several rules which transform a set of non-semantic HTML tags into a set of semantic XML tags. A semantic generator S_g is able to transform a set of HTML structures into the desired XML format. A S_g can be described like a non-empty set of $HTML_2XML$ rules that verify the format shown in expression 1.

$$HTML_2XML_i =< header > IS < body > \#num \quad (1)$$

The *header* of the rule represents the HTML tag that will be preprocessed. For instance, if the tag <table> appears in the header of the rule, all the tokens between this tag and the closing token > will be removed (so the parameters inside this tag and their values will be ignored). Moreover, all the HTML tags inserted between the tags <table> and < /table> will be ignored too. For instance, the next HTML code, that represents a table cell:

```
<TR>
  <TD>
   <A href="javascript:sg(6678)"> <B> Madrid </B> </A>
  </TD>
</TR>
```

will be represented in the header of a $HTML_2XML$ rule as: $< table.tr.td >$ **IS** $< My-XML-tag >$. And the next XML instance will be obtained: <**My-tag**> **Madrid** < /**My-XML-tag**>. When this cell is processed, the $< tr >$, $< td >$, $< Ahref... >$, and the $< B >$ tags, will be removed and only the textual information "Madrid" will be considered. The *body* of the rule corresponds to the XML (semantic) tag that will be used to translate the document. And finally, the parameter #num provides the number of HTML tags (in this example the number of possible cells with the same structure) that are affected by this rule (all the $HTML_2XML$

[2]You can obtain a complete description of them at http://www.wifo.uni-mannheim.de/~kuhlins/wrappertools/

rules have a predefined value of 1). Figure 1 shows some examples of $HTML_2XML$ rules, that belong to a semantic generator which is able to transform some table cells. The S_g can be applied in a particular Web page, or to a subset of pages with the same structure. The $HTML_2XML$ rules structure can be used to identify nested structures, so it is possible through the utilization of these rules to transform nested tables, or lists, into a more understable (and easier to manage) structures.

```
<table> IS <table-countries> #1
 <table.tr> IS <data-record> #4
   <table.tr.td> IS <country> #4
```

Figure 1: Some $HTML_2XML$ rules extracted from a Semantic Generator

3 The WebMantic approach

We have implemented an application, called WebMantic, that is able to automatically obtain the Semantic Generators and to translate a HTML page into a XML document using the related S_g. Actually, WebMantic is related with the transformation of only two types of structures, tables and lists. These two structures have been selected because they represent the most usual method to represent information in the Web pages. Figure 2 shows the architecture of WebMantic. WebMantic modules are:

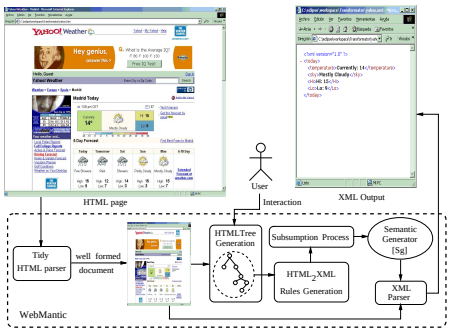

Figure 2: The WebMantic architecture.

- *Tidy HTML parser* (http://tidy.sourceforge.net). It fixes well-formed HTML documents (unbalanced tags, etc.). The HTML Tidy program (HTML parser and pretty printer) has been integrated as the first preprocessing module in WebMantic. It corrects common markup errors and outputs well formed HTML documents.
- *Tree generator module.* Once the HTML page is preprocessed by Tidy parser, a tree representation of the structures stored in the page is built. In this representation any table or list tags generate a node, and the leafs of the tree are: cells for tables (th,td,tr) or items for lists (li,lo).
- *$HTML_2XML$: Rule generator module.* The tree representation obtained is used by this module to generate a set of rules (S_g) that represent the information to be translated and what structures inside the page will be ignored, and the XML tags that have been defined by the user (or loaded directly from a file that stores the XML tags). Using both, the HTML tree and the XML tags, this module generates the following $HTML_2XML$ rules:

```
<table> IS <table-countries>
<table.tr> IS <>
<table.tr.th> IS <>
<table.tr> IS <data-record>
<table.tr.td> IS <country>
...
```

- *Subsumption module.* The previous module generates a rule for each structure to be transformed. However, some of those rules can be generalized if the XML-tag represents the same concept. (i.e. the rules in previous example that represent the concepts of <data-record> and <country>). The minimum set of rules necessary to transform the document is obtained:

```
<table> IS <table-countries>
<table.tr> IS void
<table.tr.th> IS void
<table.tr> IS <data-record> #4
<table.tr.td> IS <country> #4
```

- *XML Parser module.* This module receives both, the Semantic Generator obtained in previous module, and the (well formed) HTML document. This module generates the XML document as follows: the header of the XML document is automatically written in a file; the rules of the Semantic Generator are sequentially executed to preprocess each structure inside the HTML document.

The following (simple) example shows how a Semantic Generator is able to provide semantics generating a new XML document from an HTML document. Figure 3 shows the output, and HTML code, of the table that will be analyzed.

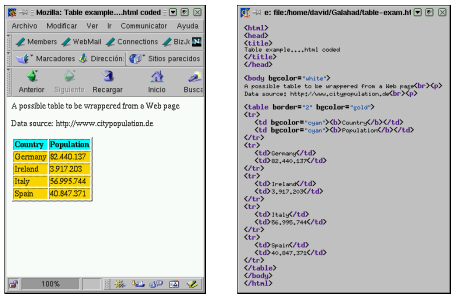

Figure 3: Table example to be transformed.

For the previous table, the user's target is to obtain the following XML representation:

```
<table-countries>
 <data-header>
   <header-country>Country</header-country>
   <header-popul>Population</header-popul>
 </data-header>
 <data-record>
   <country>Germany</country>
   <population>82.440.137</population>
 </data-record>
    <data-record>
        <country>Ireland</country>
        <population>3.917.203</population>
    </data-record>
    <data-record>
        <country>Ireland</country>
        <population>3.917.203</population>
    </data-record> ...
</table-countries>
```

Initially, using the previous HTML document WebMantic processes sequentially the HTML document and, for each structure (list or table), it asks the user for their XML tags. The next set of rules will be generated:

```
TABLE: (Sg1)                              TABLE: (Sg2)
<table> IS <table-countries> #1           <table> IS <table-countries> #1
<table.tr> IS <data-header> #1             <table.tr> IS <data-header> #1
 <table.tr.td> IS <header-country> #1       <table.tr.td> IS <header-country> #1
 <table.tr.td> IS <header-popul> #1         <table.tr.td> IS <header-popul> #1
<table.tr> IS <data-record> #1            <table.tr> IS <data-record> #4
 <table.tr.td> IS <country> #1              <table.tr.td> IS <country> #4
 <table.tr.td> IS <population> #1.....      <table.tr.td> IS <population> #4
```

The Semantic generator (S_{g1}) generated is responsible to transform the information. To translate this document it is necessary to build a semantic generator made of sixteen rules. Once the subsumption algorithm is carried out by WebMantic, the previous sets of rules are simplified to (S_{g2}). This simple example shows how the subsumption process allows to obtain the minimum number of rules necessary to build a new XML document which only stores the target information. Some of the $HTML_2XML$ rules can have a value "void", the parameter #1 shows that the next structure will be removed. In the subsumption process, if more than one (consecutive) structures are to be removed this parameter is updated appropriately.

4 Conclusions

This paper has presented an approach to wrapping semi-structured Web pages, the interaction with the user allows to build up XML-based wrappers. The main contribution of this paper is to define a technique which is able to provide a semantic representation (using XML-tags) to semi-structured (tables and lists) Web pages through a set of rules (encapsulated in a Semantic Generator). The generation of this rules is done by WebMantic. WebMantic allows the user to select portions of HTML pages, and give the semantics for them. Then, rules are created and automatically generalized. These rules can be used to preprocess Web pages with a similar structure, and convert them into XML documents with semantic tags.

Acknowledgments

This work has been funded by the Universidad de Alcalá project UAH PI2005/084.

References

[1] Serge Abiteboul, Dan Suciu, and Peter Buneman. *Data on the Web : From Relations to Semistructured Data and XML*. Morgan Kaufmann, Los Altos, CA, 1999.

[2] Robert Baumgartner, Sergio Flesca, and Georg Gottlob. Visual web information extraction with lixto. *The VLDB Journal*, pages 119–128, 2001.

[3] Yizhong Fan and Susan Gauch. Adaptive agents for information gathering from multiple, distributed information sources. In *Proceedings of 1999 AAAI Symposium on Intelligent Agents in Cyberspace*. Stanford University, March 1999.

[4] J.Y.-J. Hsu and W.-T. Yih. Template-based information mining from html documents. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 256–262. AAAI Press, Menlo Park, CA, 1997.

[5] Nicholas Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1-2):15–68, 2000.

[6] Jaideep Srivastava. Web mining - accomplishments & future directions. In *Proceedings of The Seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-03)*. Lecture Notes In Computer Science, Springer-Verlag, 2003.